Приложение к СЗ: Гриф: н/с

Экз. №: _____

ПАО «Компания «Сухой» «ОКБ Сухого»

Тема: СКТ

Техническая справка

по графику 8888.00.0019-0019/001

«Разработка модели сверточной нейронной сети, обученную на сгенерированных размеченных данных СКТ»

Начальник бр. ЭММ НИЦ СКТ И.Д. Танненберг

Инженер И.А. Ниженко

Главный конструктор СКТ А.В. Корнев

Москва 2020 г.

Содержание

1. Постановка задачи

1.1. Набор данных

Как и в случае с любой другой задачей глубокого обучения, первая важная задача - подготовить обучающий набор данных. Набор данных - это топливо, на котором работает любая модель глубокого обучения.

1.2. Алгоритм Yolo и его архитектура

1.3. Обучение

Трансферное обучение - это метод повторного использования уже предварительно обученной модели для решения новой проблемы. В настоящее время он очень популярен в глубоком обучении, потому что может обучать глубокие нейронные сети со сравнительно небольшим объемом данных и за гораздо более короткое время. Это очень полезно, поскольку большинство реальных проблем обычно не имеют миллионов помеченных точек данных для обучения таких сложных моделей.

При трансферном обучении знания уже обученной модели машинного обучения применяются к другой, но связанной проблеме. Например, если вы обучили простой классификатор предсказать, есть ли на изображении автомобиль, вы можете использовать знания, полученные моделью во время обучения, для распознавания других объектов, например грузовика.

С трансферным обучением в основном пытаются использовать то, что было изучено в одной задаче, для улучшения обобщения в другой. Переносятся веса, полученные сетью в «задаче А», в новую «задачу В.»

Общая идея состоит в том, чтобы использовать знания, полученные моделью из задачи с большим количеством доступных помеченных обучающих данных, в новой задаче, в которой не так много данных.

Трансферное обучение в основном используется в задачах компьютерного зрения и обработки естественного языка, таких как анализ тональности из-за того, что требуется огромная вычислительная мощность.

Трансферное обучение стало довольно популярным в сочетании с нейронными сетями, которые требуют огромных объемов данных и вычислительной мощности.

2. Решение задачи

2.1. Структура программного кода

```
#Blender автоматизация

  import bpy, math
  import numpy
  from numpy import genfromtxt

  scene = bpy.context.scene

  trajectory = genfromtxt('D:\\!IAN_WORK\\NN_vision\\trajectory.csv',
 delimiter=',')

  ob1 = bpy.data.objects["ASP2"]
  ob2 = bpy.data.objects["Box"]
  frame_number = 40
  ob1.animation_data_clear()
  ob2.animation_data_clear()

  positions = trajectory[:, :3]
  angles = trajectory[:, 3:]

  for angle_i, i in enumerate(positions):

    bpy.context.scene.frame_set(frame_number)

    ob1.location = i
    ob1.rotation_euler = angles[angle_i]
    ob1.keyframe_insert(data_path='location', index=-1)
    ob1.keyframe_insert(data_path='rotation_euler', index=-1)

    ob2.location = i
    ob2.rotation_euler = angles[angle_i]
    ob2.keyframe_insert(data_path='location', index=-1)
    ob2.keyframe_insert(data_path='rotation_euler', index=-1)

    frame_number += 0.1

  file = open("D:\\!IAN_WORK\\NN_vision\\trajectory.txt", 'w')
  loc = ob1.location

  for frames in range(scene.frame_start, scene.frame_end + 1):
  scene.frame_set(frames)
  file.write(str(loc.x) + ', ' + str(loc.y) + ', ' + str(loc.z) + '\n')

  file.close()

  bpy.context.scene.render.filepath = 'D:\\!IAN_WORK\\NN_vision\\RenderAll\\
 video\\'
  bpy.ops.render.render(animation=True, use_viewport=True)
  bpy.ops.render.render(write_still=True)

  bpy.data.scenes[0].filepath = 'D:\\!IAN_WORK\\NN_vision\\RenderAll\\'
  bpy.ops.render.render(animation = True)

```

```python
#Скрипт для создания коллажа

  import cv2
  from PIL import Image,ImageFont,ImageDraw
  import os

  def chunks(lst,chunk_size,step):
    for i in range(0,len(lst),chunk_size*step):
      yield  lst[i:i+chunk_size*step:step]

  def create_collage(width, height, listofimages):
    cols = 3
    rows = 3
    thumbnail_width = width//cols
    thumbnail_height = height//rows
    size = thumbnail_width, thumbnail_height
    new_im = Image.new('RGB', (width, height),color='white')
    ims = []
    for im in listofimages:
      im.thumbnail(size)
      ims.append(im)
      i = 0
      x = 0
      y = 0
      for row in range(rows):
        for col in range(cols):
          print(i, x, y)
          if i>=len(ims):
            break
          new_im.paste(ims[i], (x, y))
          i += 1
          x += thumbnail_width
          y += thumbnail_height
          x = 0
    return new_im

  def create_collage_from_video(videoname,chunk_size=9,step=15):
    basename = os.path.splitext(videoname)[0]
    cap = cv2.VideoCapture(videoname)
    images=[]
    while(cap.isOpened()):
    ret,frame=cap.read()
    if not ret:
      break
    rgb = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
    im_pil=Image.fromarray(rgb)
    images.append(im_pil)
    cap.release()
    cv2.destroyAllWindows()
    font = ImageFont.truetype('verdana.ttf',60)
    for index,image in enumerate(images):
      images[index] = image.crop((68,60,1515,914))
      draw = ImageDraw.Draw(images[index])
      draw.text((0,0),u'Кадр: {:.3f} сек.'.format(index),(0,0,0),font=font)


  image_list = chunks(images[step:],chunk_size,step).__next__()
  create_collage(1920,1200,image_list).save('{}.png'.format(basename.replace('
 .','_')))

```

3. Выводы

A. Приложение

Научно-исследовательский центр
суперкомпьютерных технологий

от 31 декабря 2020 г.

Заместителю технического
директора
Никитушкину М.В.

Служебная записка

Разработана модель сверточной нейронной сети, обученную на сгенерированных данных СКТ, для последующего применения в задаче нейронной сети в соответствии с графиком работ 8888.00.0019-0019/001

Приложение:

– Модель нейронной сети расположена на АРМ okb-26341 в директории:
\\okb—26341\External share\Finished Works генератор\ изображений\

Инженер Ниженко И.А.

Исполнитель: Ниженко И.А.
тел.: 75-39